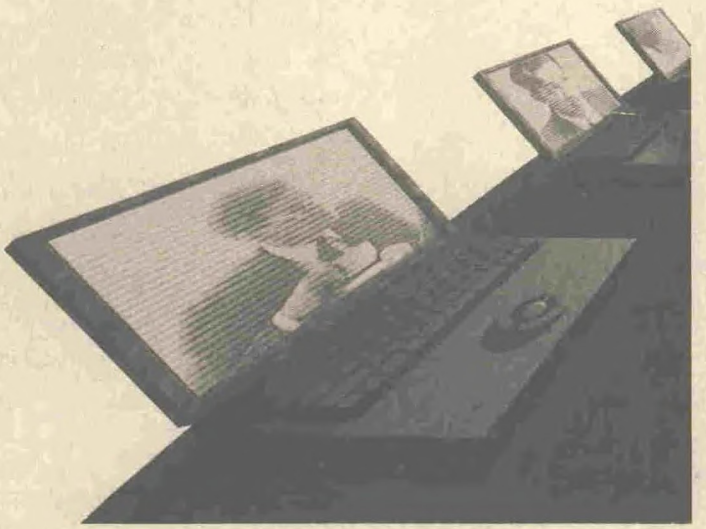
ชูเกียรติ วรสุชีพ

Rapid Application Development



เป็นเพียงแนวทางปฏิบัติคร่าว ๆ ที่น่าระลึกถึงเมื่อจะทำโครงการใด ๆ และแนวทางนี้ยังต้องอาศัยการปรับเปลี่ยนพัฒนาให้ดีขึ้นอีกต่อไป

Rapid Application Development หรือ RAD หมายถึง การพัฒนาสร้างสรรค์ระบบสารสนเทศภายใต้ระยะเวลา งบประมาณ และทรัพยากรที่มีไม่มากนัก แตกต่างจากโครงการระบบสารสนเทศขนาดใหญ่ ๆ ที่อาจยาวนานถึง 1-2 ปี หรือกว่านั้น ที่ว่าใช้ทรัพยากรไม่มากนัก ก็อย่างเช่นมีทีมงานไม่เกิน 8-9 คน ซึ่งโดยทั่วไปในเมืองไทยแล้วโครงการระบบสารสนเทศมากมายสามารถจัดเรียกได้ว่าเป็น RAD ในที่นี้ผมจะขอกล่าวถึงขั้นตอนและเทคนิคบางอย่างสำหรับการทำโครงการประเภทนี้อย่างเป็นระบบระเบียบ มีขั้นมีตอนและวัตถุประสงค์ที่ชัดเจน ซึ่งเรามักพบว่าคนไทยเรายังไม่ค่อยมีการทำงานหรือคิดอย่างเป็นแบบแผนมากนักเหมือนอย่างที่คนอเมริกันเขาเป็นกัน ซึ่งน่าจะมีส่วนช่วยทำให้เขาเจริญ (ทางวัตถุและธุรกิจ) อย่างมาก เนื้อหาของบทความนี้อาจจะค่อนข้างหนักและเป็นวิชาการไปบ้าง แต่อย่าเบื่อเลยครับ หลักการดี ๆ มีประโยชน์ก็มักจะไม่ค่อยน่าอ่านเท่าไร

วัตถุประสงค์หลักของโครงการที่เป็น RAD มี 4-5 ข้อ ได้แก่

1. เพื่อพัฒนาระบบสารสนเทศ (Information System) ให้แก่องค์กร (ธุรกิจ) ภายในระยะเวลาที่สั้นกว่าการพัฒนาระบบทั้งหมดที่จะใช้สำหรับองค์กรนั้น

2. เพื่อพัฒนาระบบสารสนเทศในลักษณะเพิ่มเติมและวนซ้ำ (Incremental and Interative)

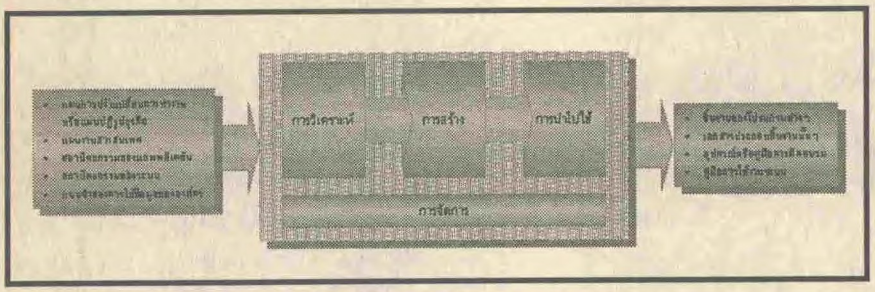
3. เพื่อรองรับการพัฒนาภายใต้งบประมาณและเวลาจำกัดและน้อย

4. เพื่อรองรับงานสำหรับทีมงานขนาดเล็ก (4-7 คน) ที่ทำโครงงานขนาดประมาณ 3 ถึง 6 เดือน และต้องสื่อสารสัมพันธ์กับผู้ใช้ระบบอย่างมากในขั้นตอนการพัฒนา เพื่อให้สอดคล้องกับความต้องการใช้ระบบ

ปกติแล้วเวลาที่เราทำโครงการ RAD ก็ควรอาศัยเทคนิคการพัฒนาระบบ ผู้ใช้ เครื่องมือ และเทคนิคการบริหารโครงการอย่างมีระเบียบแบบแผน เพื่อจะสร้างระบบสารสนเทศที่ต้องการภายในระยะเวลาที่กำหนดภายใต้ทรัพยากรที่จำกัด RAD ได้ย่อส่วนวิธีการพัฒนาระบบขนาดใหญ่ทั่วไปลงให้เหลือเพียงส่วนที่จำเป็นในการพัฒนาระบบสารสนเทศ เราสามารถแบ่งขั้นตอนออกเป็น 3 ส่วนใหญ่ ๆ ต่อเนื่องกันคือ การวิเคราะห์ (Analysis) การสร้าง (Development) และการนำไปใช้ (Deployment)

โครงการ RAD มุ่งเน้นกระบวนการนำความต้องการใช้ระบบ (System Requirement) เข้าสู่ขั้นตอนการสร้างซึ่งอาจจะวนซ้ำ (Interation) การวนซ้ำจะอยู่ในกรอบเวลาซึ่งเราต้องควบคุมการใช้ทรัพยากรที่มีสำหรับในแต่ละรอบการวนซ้ำนี้ ท้ายที่สุดจะได้มาซึ่งระบบสารสนเทศที่จะนำไปใช้จริง

นอกจากสามขั้นตอน (การวิเคราะห์ การสร้าง และการนำไปใช้) แล้ว ยังมีส่วนสำคัญอีกส่วนหนึ่งที่ขาดไม่ได้คือ การจัดการ (Manage) ที่จะยังคงอยู่ตลอดช่วงของโครงการ ดังในรูปที่ 1



รูปที่ 1 โครงสร้างโดยรวมและสิ่งที่เป็นอินพุตและผลลัพธ์จากโครงการแบบ Rapid Application Development

1. การวิเคราะห์ วัตถุประสงค์ก็เพื่อกำหนดและจัดลำดับความต้องการของธุรกิจทั้งส่วนที่จำเป็นและเกี่ยวเนื่อง ขั้นตอนนี้ประกอบด้วยงานและเทคนิคที่เกี่ยวข้องกับการกำหนดและจัดลำดับความต้องการใช้ระบบ ซึ่งจะกำหนดหน้าที่และขอบเขตของระบบ

2. การสร้าง โครงการแบบ RAD อาศัยกระบวนการวนซ้ำในการออกแบบ เขียน และทดสอบ ในส่วนของการสร้าง แต่ละรอบการวนซ้ำจะอยู่ภายในระยะเวลาที่กำหนด เพื่อจะได้สร้างเพิ่มเติมเป็นขั้น ๆ ไป แทนที่จะทำให้เสร็จในคราวเดียว (Big Bang) ในการสร้างนี้จะคำนึงถึงความต้องการของระบบ ข้อคิดเห็นและข้อแนะนำของผู้ใช้เข้ามาด้วย

3. การนำไปใช้ ส่วนของการนำไปใช้จะต่อจากการสร้าง โดยที่จะมีการสร้างขั้นตอนการทำงาน ใช้งานและการฝึกอบรมให้แก่ผู้ใช้ รวมทั้งการแปลงข้อมูลเก่าที่อาจมีอยู่มาเข้าสู่ระบบใหม่ที่พัฒนาขึ้นมา

สิ่งที่ต้องการและผลลัพธ์

จากรูปที่ 1 สิ่งที่เราต้องใช้เป็น input ในการทำโครงการหรือโปรเจ็กต์ประเภท RAD ได้แก่ แผนการปรับเปลี่ยนการทำงานหรือแผนปฏิรูปธุรกิจ แผนงานสารสนเทศ สถาปัตยกรรมของแอพพลิเคชัน สถาปัตยกรรมของระบบ และแบบจำลองการใช้ข้อมูลขององค์กร

แผนการปรับเปลี่ยนการทำงาน (Business Reengineering Plan) จะกล่าวถึงกลยุทธ์ แนวนโยบาย และองค์ประกอบระยะยาว (หลายปี) สำหรับการปรับปรุงการทำงานในองค์กร สถาปัตยกรรมของระบบจะเป็นร่างคร่าว ๆ ของระบบใหม่ที่จะนำมาใช้โดยแสดงขอบเขตความสัมพันธ์ต่อองค์ประกอบของระบบอื่น ๆ แผนงานสารสนเทศหรือ Information Plan จะเป็นแผนข้อมูลสำหรับโครงการพัฒนาระบบงานต่าง ๆ ในช่วงระยะเวลาหนึ่งอาจจะ 2-4 ปี ซึ่งจะครอบคลุมแผนการนำเทคโนโลยีมาใช้ และอาจรวมทั้งแผนการจัดการโครงสร้างองค์กรด้วย สถาปัตยกรรมของแอพพลิเคชันจะบอกความสัมพันธ์ระหว่างการทำงานส่วนต่าง ๆ ขององค์กรนั้นกับแอพพลิเคชันต่าง ๆ ที่จะเลือกใช้ทั้งที่ทำเอง (custom built หรือ in-house developed) และซื้อมาประยุกต์หรือปรับใช้ (customized) แบบจำลองการใช้ข้อมูลขององค์กร (Enterprise Data Model) จะแสดงองค์ประกอบข้อมูลต่าง ๆ และความสัมพันธ์ต่อองค์กรโดยรวม แบบจำลองการใช้ข้อมูลขององค์กรจะไม่ได้แสดงในรายละเอียด เพียงแต่ใช้สื่อสารให้เข้าใจแผนงานโดยรวม และเป็นองค์ประกอบสำคัญสำหรับโครงการย่อย ๆ ทั้งหลาย

สิ่งที่เราจะได้จากโครงการประเภท RAD ได้แก่ ชิ้นงานของโปรแกรมต่าง ๆ เอกสารประกอบชิ้นงานนั้น ๆ อุปกรณ์หรือคู่มือการฝึกอบรม และคู่มือการใช้งานระบบที่เสร็จแล้ว ชิ้นงานต่าง ๆ ก็ได้แก่ โปรแกรมโค้ด ข้อกำหนดหรือสเปกของชิ้นงานส่วนต่าง ๆ รวมทั้งคำอธิบายส่วนต่าง ๆ เอกสารประกอบชิ้นงาน ได้แก่ คู่มือทางเทคนิคที่จะอธิบายให้ทราบเกี่ยวกับชิ้นงานนั้น ๆ เช่น เรื่องของฮาร์ดแวร์ ซอฟต์แวร์ การติดตั้ง และปรับปรุงรวมทั้งการปฏิบัติงานประจำวัน หรือสัปดาห์หรือเดือนแล้วแต่ และการทำสำรอง (backup) และการทดสอบสำรองที่มีอยู่นั้นซึ่งเป็นสิ่งจำเป็นที่มักถูกมองข้าม อุปกรณ์การฝึกอบรมอาจจะเป็นคู่มือให้อ่านหรือคู่มือการสอน ทั้งในรูปหนังสือ แผ่นพับหรือทางคอมพิวเตอร์ โดยควรครอบคลุมทั้งสำหรับผู้ใช้ในปัจจุบันและผู้ใช้คนใหม่ที่อาจเพิ่มขึ้นในอนาคต คู่มือการใช้งานระบบ (ทั้งในรูปของเอกสารการพิมพ์และอิเล็กทรอนิกส์) ก็ช่วยให้ผู้ใช้เข้าใจการใช้งาน รวมทั้งงานที่ต้องทำเป็นประจำ และเอกสารอ้างอิงการใช้งาน (reference manual)

ต่อไปเราจะลงไปในรายละเอียดของการวิเคราะห์ การสร้าง และการนำไปใช้งาน รวมทั้งการจัดการ ว่าแต่ละส่วนมีวัตถุประสงค์ วิธีการ สิ่งที่ต้องการ และผลลัพธ์เป็นอย่างไร

1. การวิเคราะห์

วัตถุประสงค์ของส่วน "การวิเคราะห์" ก็เพื่อ

1. กำหนดและจัดลำดับความต้องการแอพพลิเคชันในระดับหนึ่งเพื่อวางแผนการสร้างในส่วนถัดไปและการจัดสรรทรัพยากรเพื่อตรวจสอบขั้นตอนหรืองานของธุรกิจนั้น ๆ และเพื่อวางมาตรฐานในการตรวจสอบแอพพลิเคชันที่จะสร้างขึ้น

2. เพื่อตกลงกับผู้สนับสนุนโครงการและตัวแผนของผู้ใช้งาน ในเรื่องของความต้องการใช้งานของระบบและกำหนดขอบเขตของงานและสร้างความเข้าใจร่วมกันเกี่ยวกับโครงงานนี้

วิธีการของส่วนการวิเคราะห์จะมุ่งเน้นการกำหนดและลำดับความต้องการของระบบโดยจะสร้างและร่างขอบเขตความต้องการและจัดลำดับความสำคัญเพื่อจะได้กำหนดขอบเขตของส่วนการสร้างถัดไปได้โดยไม่ต้องเขียนสเปกของระบบอย่างละเอียดและครบถ้วนอย่างที่โครงการขนาดใหญ่จะต้องทำ แต่ควรจะบันทึกรายละเอียดระดับหนึ่งซึ่งเพียงพอจะให้เกิดภาพรวมของระบบเพื่อนำไปใช้ในขั้นตอนการสร้างถัดไปได้

จากรูปที่ 2 จะเห็นได้ว่า ช่วงแรกของส่วนการวิเคราะห์จะเป็นการกำหนดความต้องการของระบบ โดยการ

1. กำหนดความต้องการของผู้ใช้งาน
2. กำหนดมาตรฐานและเป้าหมาย
3. สร้างแบบจำลองสถานการณ์การใช้งาน (Event Model)

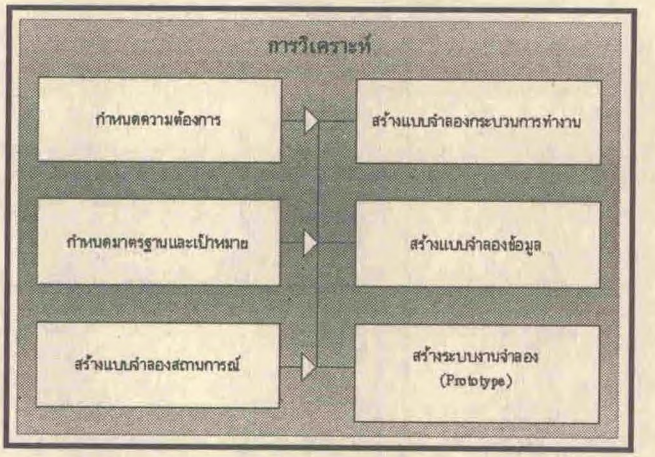
ช่วงที่สองของส่วนการวิเคราะห์จะเกี่ยวข้องกับการ

1. สร้างแบบจำลองกระบวนการทำงาน (Process Model)
2. สร้างแบบจำลองของข้อมูล (Data Model)
3. สร้างระบบงานจำลอง (Prototype) ของระบบงานนั้น ๆ เพื่อให้เห็นภาพจำลองของการใช้งานที่จะเกิดขึ้นจริงภายหลัง

ก่อนที่กล่าวถึงรายละเอียดทั้ง 6 ช่วงของส่วนการวิเคราะห์นี้ จะขอกล่าวถึงสิ่งที่ต้องการและผลลัพธ์ที่ได้จากส่วนการวิเคราะห์ก่อน สิ่งที่เราต้องการก่อนจะทำการวิเคราะห์นี้ได้แก่

1. แบบร่างของสถาปัตยกรรมของระบบที่จะแสดงขอบเขตและความสัมพันธ์ขององค์ประกอบหลัก ๆ ของระบบ
2. เอกสารข้อมูลของระบบที่ปัจจุบันใช้อยู่ เพื่อใช้ประโยชน์จากสิ่งที่มีอยู่และใช้วิเคราะห์การเชื่อมต่อ
3. แผนสารสนเทศ
4. แบบจำลองการใช้ข้อมูลขององค์กร ซึ่งกล่าวไปแล้วข้างต้น

ผลลัพธ์ที่ได้จากการทำงานส่วนการวิเคราะห์นี้ได้แก่ เอกสารที่บอกความต้องการของผู้ใช้ ระดับคุณภาพที่ผู้ใช้คาดหวัง แบบจำลองข้อมูล แบบจำลองสถานการณ์การใช้งาน และแบบจำลองกระบวนการทำงาน



รูปที่ 2
ส่วนประกอบของงานส่วน "การวิเคราะห์"

1.1 กำหนดความต้องการของผู้ใช้งาน

วัตถุประสงค์เพื่อเข้าใจความต้องการด้านการใช้งานและคุณภาพของระบบที่จะสร้างขึ้น และเพื่อตกลงถึงความต้องการทางธุรกิจและลำดับความสำคัญ

สิ่งสำคัญอย่างยิ่งอันหนึ่งคือ ความร่วมมือของผู้ใช้ระบบ ควรจะต้องมีตัวแทนของผู้ใช้งานที่จะร่วมกับทีมงานในการตัดสินใจในเรื่องความต้องการและการจัดลำดับความสำคัญต่าง ๆ

ในขั้นตอนนี้มีหลักการที่ควรคำนึงถึงได้แก่

1. เนื่องจากวัตถุประสงค์คือการรวบรวมความต้องการไม่ใช่การออกแบบ ดังนั้นเราจึงควรมุ่งเน้นไปที่ความจำเป็นของธุรกิจ ปัญหาหรือโอกาสมากกว่า
2. คำนึงถึงคุณภาพของงานที่ต้องการด้วยในการจัดลำดับความสำคัญของความจำเป็นทางธุรกิจและความต้องการด้านการใช้งาน
3. ทีมงานจะต้องรีบกำหนดและทำความเข้าใจในขอบเขตการทำงานของทีม

ในขั้นตอน "กำหนดความต้องการของผู้ใช้งานนี้" อาจเป็นการพูดคุยในเบื้องต้น การใช้แบบสอบถาม และต่อด้วยการสัมภาษณ์ (ทั้งแบบเดี่ยวหรือแบบกลุ่ม) หากสัมภาษณ์แล้วยังไม่ชัดเจนพออาจมีรอบสัมภาษณ์เก็บตกอีกครั้งหนึ่ง การแบ่งกลุ่มสัมภาษณ์นี้แต่ละรอบควรใช้เวลาไม่นานนัก เช่น 2 วัน ซึ่งมักต่างกับโครงการขนาดใหญ่ซึ่งอาจมีกลุ่มให้สัมภาษณ์มากมายและนานถึง 2 สัปดาห์หรือมากกว่า หลังจากรวบรวมข้อมูลแต่ละรอบได้ก็ต้องมาทบทวนและวิเคราะห์ผลที่ได้ และอาจต้องพูดคุยกับผู้ใช้บางคนเพื่อแก้ข้อขัดแย้งหรือสิ่งที่คลุมเครืออยู่ให้หมดไป

เรื่องของคุณภาพ ทีมงานต้องวิเคราะห์ถึงความจำเป็นทางธุรกิจ ปัญหาและโอกาสอย่างมีหลักการและวิเคราะห์ถึงความสัมพันธ์ระหว่างความจำเป็นทางธุรกิจกับความต้องการด้านการใช้งาน

การเลือกผู้ที่เราจะไปสัมภาษณ์หรือให้ตอบแบบสอบถามก็เป็นสิ่งสำคัญ ผู้ถูกเลือกต้องมีอำนาจหน้าที่และเข้าใจระบบงานที่มีอยู่อย่างแท้จริง เพื่อจะได้ร่วมกันกำหนดและจัดลำดับความสำคัญของความต้องการได้ถูกต้อง ผลที่ได้จากขั้นตอนนี้ก็จะต้องนำไปจัดลำดับและบันทึกเป็นรายงานขึ้นมาอย่างมีระเบียบแบบแผน เข้าใจง่ายและครบถ้วน การบันทึกก็ต้องบันทึกทั้งความต้องการทั้งเชิงการใช้งานและเชิงคุณภาพ เชิงการใช้งานหมายถึงแอพพลิเคชันจะทำอะไร มีอะไรเป็นอินพุตและเอาต์พุต เป็นต้น ส่วนเชิงคุณภาพจะพิจารณาว่าแอพพลิเคชันนั้นทำงานอย่างไร ซึ่งจะเป็นตัวตัดสินเมื่อมีคำถามว่าจะเลือกทางไหนเมื่อมีค่าใช้จ่ายเท่ากัน ผู้ออกแบบจะต้องเลือกวิถีทางที่ให้ได้คุณภาพสูงที่สุด เป็นต้น

1.2 กำหนดมาตรฐานและเป้าหมาย

วัตถุประสงค์เพื่อชี้แจงความต้องการเชิงคุณภาพในลักษณะที่ตรวจวัดได้ วิธีการก็จะต้องกำหนดมาตรวัด และตามด้วยการกำหนดเกณฑ์มาตรฐาน (benchmark) การกำหนดมาตรวัดก็โดยการแบ่งคุณสมบัติด้านคุณภาพ

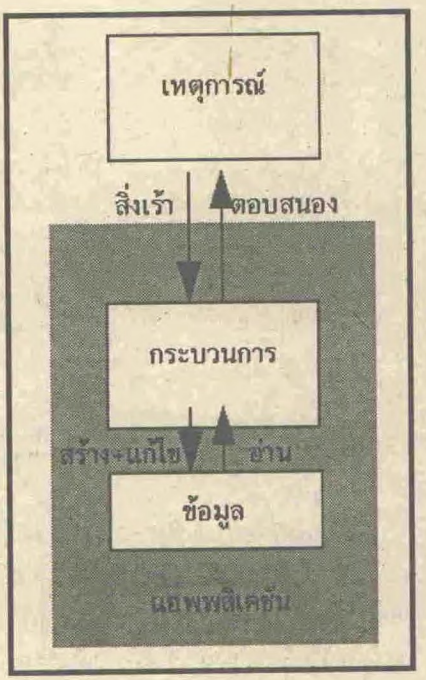
ออกเป็นหน่วยย่อย ๆ ที่วัดได้ กำหนดหน่วยวัดและกำหนดวิธีการวัด แต่ละหน่วยย่อยนั้น ๆ การกำหนดเกณฑ์มาตรฐานก็โดยการวัดระดับที่เป็นอยู่ และวัดปริมาณ ราคา และเวลาของเกณฑ์มาตรฐานนั้น ๆ

การกำหนดมาตรวัดเป็นสิ่งสำคัญ เนื่องจากเป็นเกณฑ์ในการชดเชยหรือทดแทน (tracleoff) ในขณะทำการสร้างระบบ และช่วยให้ทีมรู้ว่าเมื่อไรที่ระบบสามารถตอบสนองความต้องการต่าง ๆ ได้ ความต้องการเชิงคุณภาพจะเป็นเกณฑ์ควบคุมเปรียบเทียบอนาคตกับปัจจุบันได้เป็นอย่างดี โดยมีการตั้งเป้าว่าหลังจากใช้ระบบใหม่แล้ว งานที่ได้จะต้องเร็วขึ้นแค่ไหน มากขึ้นแค่ไหน ประหยัดทรัพยากรอะไรแค่ไหน เป็นต้น เกณฑ์มาตรฐาน (benchmark) อาจจะใช้ที่เป็นมาตรฐานสากลหรือคิดสร้างหรือประยุกต์มาก็ได้ ในบางครั้งอาจจะไม่สามารถวัดค่าปัจจุบันตามเกณฑ์ได้ ก็อาจจะคิดวิธีการเพื่อคำนวณโดยอาจใช้สถิติแบบสอบถามหรือการสุ่มเพื่อวัดค่าปัจจุบันและตั้งเป้าหมายของเกณฑ์ต่าง ๆ ในอนาคต ก็ควรจะมีผู้เชี่ยวชาญที่มีประสบการณ์ในระบบงานที่ทำนี้ เพื่อตรวจสอบระดับเป้าหมายในอนาคตที่ตั้งไว้ และอาจปรับปรุงค่าต่าง ๆ เหล่านั้นให้เหมาะสมยิ่งขึ้น

1.3 สร้างแบบจำลองเหตุการณ์ (Event Model)

วัตถุประสงค์เพื่อจำลองการตอบสนองต่อเหตุการณ์หรือที่เรียกว่าสิ่งเร้า (Stimulus) จากธุรกรรมต่อแอพพลิเคชัน และเพื่อให้เข้าใจแบบจำลองอื่น ๆ ได้ง่าย

แบบจำลองสถานการณ์อธิบายระบบที่จะมีในอนาคตในการตอบสนองต่อธุรกรรมและเวลาซึ่งจะทำให้ระบบทำงาน องค์ประกอบของแบบจำลองสถานการณ์ได้แก่ เหตุการณ์-สิ่งเร้า-การตอบสนอง ดังในรูปที่ 3 สิ่งเร้าทำให้เกิดกระบวนการทำงาน (Process) ซึ่งจะตอบสนองต่อเหตุการณ์ งานนี้เกิดภายในแอพพลิเคชันและอาจใช้ข้อมูลที่เก็บอยู่ในการตอบสนองแต่ละครั้ง



รูปที่ 3 แบบจำลองเหตุการณ์

เราสามารถใช้เหตุการณ์ในการอธิบายส่วนงานต่าง ๆ ของระบบได้ เหตุการณ์ทั้งหมดรวมกันจะบอกถึงขอบเขตของกระบวนการทั้งหมดในระบบ ซึ่งจะเริ่มต้นจากเหตุการณ์ใดเหตุการณ์หนึ่ง การใช้เหตุการณ์จะช่วยในการอธิบายการทำงานของระบบในแต่ละรอบการวนซ้ำในส่วนของการสร้าง โดยที่เหตุการณ์ในแต่ละรอบจะไม่ล้ำไปที่รอบอื่น ๆ

1.4 สร้างแบบจำลองกระบวนการทำงาน (Process Model)

วัตถุประสงค์เพื่อบรรยายพฤติกรรมทั้งหมดของแอพพลิเคชันในการตอบสนองต่อธุรกรรมและเหตุการณ์ต่าง ๆ และเพื่อตรวจสอบความต้องการการใช้งาน โดยข้อมูลจะถูกส่งให้แต่ละกระบวนการ (Process) แล้ว จะตรวจสอบข้อมูลที่ได้ออกมา

ขั้นตอนก็เริ่มจากพิจารณาวิธีการสร้างแบบจำลอง สร้างแบบจำลอง ทดสอบกับแบบจำลองทางธุรกิจอื่น ๆ (เช่น Event Model) ตรวจสอบแบบจำลองว่าตรงตามความต้องการข้อมูลและเหตุการณ์หรือไม่ และก็ต้องตรวจสอบแบบจำลองนั้นกับผู้ใช้ด้วย

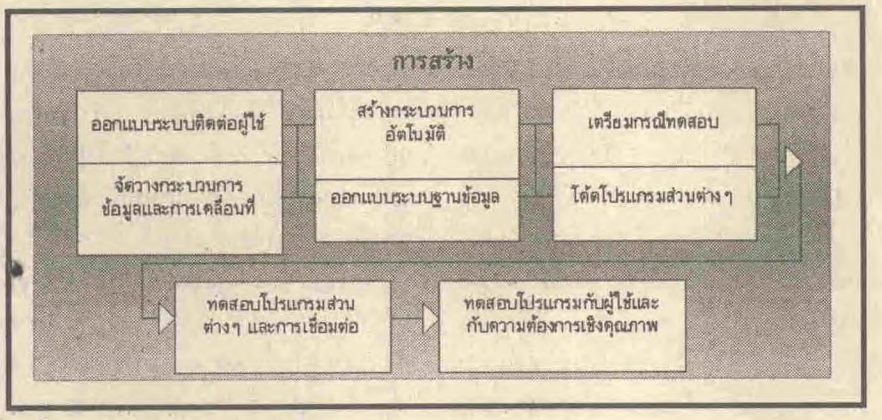
แบบจำลองกระบวนการทำงาน จะบอกถึงพฤติกรรมและโครงสร้างภายในของแอพพลิเคชัน ซึ่งถูกกำหนดโดยเหตุการณ์และข้อมูล ดังนั้นแบบจำลองทั้งสาม (กระบวนการ เหตุการณ์ และข้อมูล) จะถูกสร้างขึ้นพร้อม ๆ กัน แบบจำลองกระบวนการจะเป็นพื้นฐานสำคัญสำหรับการโค้ดโปรแกรม มันจะบอกถึงวิธีการจัดการกับข้อมูลตามความต้องการของระบบที่ได้มาก่อนหน้านี้

แบบจำลองกระบวนการทำงานอาจจะเป็น ผังการเคลื่อนที่ของข้อมูลหรือ Data Flow Diagram (DFD) หรือเป็นผังแบ่งย่อยงานธุรกรรมหรือ Business Function Decomposition (BFD)

DFD อธิบายการเคลื่อนที่ของข้อมูลต่าง ๆ จากภายนอก หรือแหล่งข้อมูล (Source) ไปยังกระบวนการ (Process) ต่าง ๆ และแหล่งข้อมูล (Data Store) อย่างเป็นขั้นเป็นตอนและออกไปยังปลายทางข้อมูลภายนอก (Sink) แต่ละการเคลื่อนที่ของข้อมูลจะมีคำอธิบายคุณลักษณะซึ่งสอดคล้องกับเหตุการณ์หรือสิ่งเร้าที่มีต่อกระบวนการนั้น ๆ

ในการสร้าง DFD มีข้อคำนึงอยู่บางประการคือ ไม่ควรลงกระบวนการหรือข้อมูลที่ขึ้นอยู่กับวิธีการสร้าง (implementation) ในภายหลัง คือไม่ผูกติดกับเทคโนโลยีใดเทคโนโลยีหนึ่ง แต่ละผังควรจบในหนึ่งหน้ากระดาษหรือหน้าจอคอมพิวเตอร์ ถ้ากระบวนการอันหนึ่งซับซ้อนหรือมีข้อมูลเข้ามากมาย ควรจะแตกย่อยมันให้เล็กลง DFD ที่ได้ต้องครบถ้วน นั่นคือแต่ละกระบวนการต้องมีข้อมูลเข้าและข้อมูลออกอย่างน้อยอย่างละหนึ่งอัน (ยกเว้น Source และ Sink)

BFD อธิบายงานธุรกรรมต่าง ๆ ของธุรกิจ โดยการแบ่งย่อยลงในลักษณะ Top Down แต่ละครั้งของการแบ่งควรมีงานย่อยสัก 4-7 งานย่อย ก็จะเหมาะสมเข้าใจง่าย ข้อคิดอีกอย่างหนึ่ง การแบ่งย่อยคือควรจะดูระดับงานของผู้อ่าน (ผู้จัดการ ผู้เชี่ยวชาญ หรือมือใหม่ เป็นต้น) และวัตถุประสงค์ด้วย



รูปที่ 4 ส่วนประกอบของงานส่วน "การสร้าง"

1.5 สร้างแบบจำลองข้อมูล (Data Model)

วัตถุประสงค์เพื่อบ่งบอกความต้องการในด้านข้อมูลอย่างครบถ้วน โดยอิสระจากกระบวนการสร้างภายหลัง และเพื่อประโยชน์ในการออกแบบฐานข้อมูลของแอพพลิเคชัน

แบบจำลองข้อมูลจะแสดงลักษณะความต้องการด้านข้อมูลและวิธีปฏิบัติต่อข้อมูลเหล่านั้น โดยอิสระจากวิธีการสร้างที่จะเกิดขึ้นภายหลัง วิธีการสร้างอาจเป็นแบบ Top Down หรือ Bottom Up โดยทั่วไปจะเป็นแบบผสม โดยมี Top Down เป็นโครงใหญ่ และใช้ Bottom Up เสริมให้เต็มหรือตรวจทาน ขั้นตอนการสร้างจะเริ่มจาก ตรวจทานแบบจำลองข้อมูลรวมขององค์กร กำหนดขอบเขต กำหนดชนิดและความสัมพันธ์ของข้อมูล กำหนดกฎการใช้งานข้อมูลต่าง ๆ ทำการ normalize ข้อมูล รวมข้อมูลบางส่วนถ้ารวมได้ ตรวจสอบความครบถ้วนของข้อมูลทั้งหมดและตรวจสอบแบบจำลองข้อมูลกับผู้ใช้

1.6 สร้างระบบงานจำลอง (Prototype)

วัตถุประสงค์เพื่อให้ผู้ใช้ได้เห็นหน้าตาคร่าว ๆ ของระบบที่จะสร้างขึ้นมาต่อไป ซึ่งจะช่วยในการสื่อสารให้ผู้ใช้เห็นภาพพจน์ได้ชัดเจนกว่าคำอธิบายบนกระดาษ ทั้งนี้ก็เพื่อตรวจสอบกับผู้ใช้ถึงความเข้าใจในงานของผู้พัฒนาระบบนั่นเอง

สิ่งที่ต้องทำก็เริ่มจากต้องระบุขอบเขตของงานจำลอง เลือกวิธีการสร้าง แล้วออกแบบงานจำลองนั้นขึ้นมา โดยเลือกเหตุการณ์ ข้อมูลที่ควรจะแสดงให้ผู้ใช้เห็นภาพลักษณ์ได้ดีที่สุด แล้วพัฒนาและทดสอบงานจำลองนั้นกับผู้ใช้

การทำงานจำลอง (Prototype) ช่วยลดความเสี่ยงของการพัฒนาลงได้ เนื่องจากจะได้มีการตรวจสอบและยืนยันความต้องการของผู้ใช้ได้ระดับหนึ่ง และช่วยยืนยันแนวทางวิธีการพัฒนาที่นำมาใช้ได้ ตั้งแต่ช่วงแรก ๆ ของโครงการด้วย เพื่อเป็นพื้นฐานสำหรับการพัฒนาระบบในช่วงถัดไป

การออกแบบงานจำลอง ควรจะเลือกระดับความลึกที่พอสมควร งานจำลองที่ง่ายจะช่วยลดการใช้ทรัพยากรทั้งคน เวลา และอื่น ๆ ลงในการเตรียม ในการเลือกชิ้นส่วนของงานที่จะมาสร้างงานจำลอง ก็ควรเลือกส่วนที่เป็นความต้องการด้านคุณภาพสูง ๆ หรือส่วนที่ผู้ใช้เข้าใจลำบาก

2. การสร้าง

วัตถุประสงค์ของส่วน "การสร้าง" (Development) ก็เพื่อออกแบบและพัฒนาแอพพลิเคชัน (โดยมักจะทำซ้ำแบบ Interation) ภายในระยะเวลาและทรัพยากรที่มีอยู่ เพื่อบรรลุความต้องการทั้งด้านการใช้งานและคุณภาพ ในช่วงนี้ทีมจะออกแบบพัฒนาและทดสอบระบบ ซึ่งทำให้พบกับความต้องการใหม่ ๆ หรือที่ละเอียดลงไปเพิ่มขึ้น

งานในส่วน "การสร้าง" นี้จะเกี่ยวข้องโดยตรงกับการพัฒนาระบบไคลเอนต์/เซอร์ฟเวอร์ใหญ่ แต่ย่อขนาดลงมา โดยอาจจะแบ่งออกเป็น 5 ขั้นตอนคร่าว ๆ ดังในรูปที่ 4 คือ

1. ออกแบบระบบติดต่อผู้ใช้ จัดวางกระบวนการ ข้อมูล และทิศทางเดินของข้อมูล
2. ออกแบบฐานข้อมูล และสร้างกระบวนการอัตโนมัติ
3. โค้ดโปรแกรมส่วนต่าง ๆ และเตรียมกรณีทดสอบ (Test Case)
4. ทดสอบโปรแกรมส่วนต่าง ๆ และการเชื่อมต่อกัน
5. ทดสอบโปรแกรมกับผู้ใช้และความต้องการเชิงคุณภาพ

ก่อนที่เราจะลงรายละเอียดของ 5 ขั้นตอนนี้ ลองไปดูสิ่งที่ต้องการและผลลัพธ์สำหรับช่วง "การสร้าง" นี้ก่อน

สิ่งที่ต้องการส่วนใหญ่ก็จะเป็นผลลัพธ์จากส่วนที่แล้ว (การวิเคราะห์) ได้แก่ แบบจำลองเหตุการณ์ แบบจำลองข้อมูล แบบจำลองกระบวนการทำงาน และงานจำลอง (Prototype) ที่สร้างขึ้น นอกจากนั้นก็ยังมีงานออกแบบการเลือกใช้แอพพลิเคชันอย่างไร หรือที่เรียกว่าสถาปัตยกรรมของแอพพลิเคชัน (Application Architecture) นั่นเอง ซึ่งแสดงภาพรวมว่าจะใช้แอพพลิเคชันใดบ้างอย่างไร เทียบกับงานธุรกรรมส่วนต่าง ๆ ทั้งหมด เราสามารถได้มาตรฐานระบบการติดต่อผู้ใช้จากสถาปัตยกรรมแอพพลิเคชันด้วย โดยเราอาจยืมส่วนนี้มาจากระบบงานที่มีอยู่แล้วเพื่อความสอดคล้อง นอกจากนี้งานส่วน "การสร้าง" นี้ก็ต้องการเอกสารที่อธิบายคุณสมบัติด้านคุณภาพและการใช้งานของระบบ วิธีการหรือแนวทางการทดสอบระบบ รวมทั้งเครื่องไม้เครื่องมือและเอกสารอธิบายทุกอย่างที่จะใช้พัฒนาระบบ

ผลลัพธ์ที่ได้จากส่วน "การสร้าง" นี้ก็คือ โค้ดของโปรแกรม สเปกของอินพุต/เอาต์พุตทั้งหลาย และสิ่งอื่น ๆ ที่จำเป็นในการติดตั้งและการใช้งานระบบนี้ รวมทั้งเอกสารอธิบายวิธีการออกแบบด้วย

2.1 ออกแบบระบบติดต่อผู้ใช้ จัดวางกระบวนการ ข้อมูล และทิศทางเดินของข้อมูล

วัตถุประสงค์ในการออกแบบระบบติดต่อผู้ใช้ก็เพื่อออกแบบส่วนที่ผู้ใช้จะพบเห็นโดยตรงในการใช้งาน เช่น หน้าต่าง หรือข้อความบนจอภาพ รวมทั้งเอกสารต่าง ๆ ก็เพื่อยืนยันต่อความต้องการทั้งด้านการใช้งาน คุณภาพ และความน่าเชื่อถือ เช่น การควบคุมต่าง ๆ

วิธีการออกแบบระบบติดต่อผู้ใช้ ก็เริ่มจากทบทวนวัตถุต่าง ๆ ที่จะมี ออกแบบการเชื่อมต่อของหน้าจอหรือหน้าต่าง (Windows and Dialog Box) ว่าจอไหนไปจอไหนเมื่อไร (เกิดเหตุการณ์อะไรขึ้นและ/หรือภายใต้เงื่อนไขอะไร) จากนั้นออกแบบหน้าตาและองค์ประกอบของแต่ละหน้าต่าง แล้วทบทวนดูทั้งหมดอีกครั้ง แล้วออกแบบรายงานและเอกสาร แล้วจึงไปทบทวนกับผู้ใช้งาน

วัตถุประสงค์การจัดวางกระบวนการข้อมูล และทิศทางเดินของข้อมูล ก็เพื่อกำหนดโปรแกรมและข้อมูลทั้งหมดสำหรับแอพพลิเคชัน โดยพิจารณาถึงลำดับและการเชื่อมต่อประสานกันของข้อมูลด้วย

วิธีการคือจัดวางงานหรือกระบวนการต่าง ๆ และความสัมพันธ์ กำหนดข้อมูลที่ใช้ร่วมกันระหว่างกระบวนการต่าง ๆ ทบทวนผังที่ได้มานี้ กำหนดข้อมูลที่ใช้ภายในแต่ละกระบวนการ กำหนดลำดับและการสื่อสารระหว่างโปรแกรมต่าง ๆ กำหนดส่วนเชื่อมต่อ (interface) กับระบบอื่น ๆ (ถ้ามี) กำหนดการควบคุมภายในแต่ละโปรแกรมและระหว่างโปรแกรม กำหนดออกแบบโปรแกรมประกอบอื่น ๆ เช่น โปรแกรมที่ช่วยด้านความปลอดภัยหรือจัดการฐานข้อมูล จากนั้นจึงพิจารณาดูว่าส่วนไหนพอจะใช้ร่วมกันได้เพื่อประโยชน์ในการนำมาใช้ใหม่ (reuse)

โปรแกรมแต่ละโปรแกรมอาจจะเป็นแบบออนไลน์ (ไคลเอนต์) หรือเป็นเซอร์ฟเวอร์ แบบออนไลน์มักจะต้องคำนึงอย่างมากเกี่ยวกับการออกแบบหน้าต่างและการเชื่อมต่อ ส่วนแบบเซอร์ฟเวอร์มักต้องคำนึงมากถึงเรื่องของข้อมูลและบริการที่จะมีให้ไคลเอนต์ใช้งาน

หากโปรแกรมเป็นแบบแบตช์ (Batch) งานย่อย ๆ ควรจะรวมกันเป็นหนึ่งโปรแกรมแบตช์ เพื่อลดค่าโสหุ้ยหรือโอเวอร์เฮดในการเรียกโปรแกรม และลดความวุ่นวายจากการมีมากโปรแกรมเกินไป ขณะเดียวกันก็ไม่ควรจะรวมมากงานเกินไปจนกลายเป็นโปรแกรมแบตช์ที่ใหญ่เกินไป เพื่อลดความซับซ้อนของโปรแกรม

การกำหนดส่วนควบคุมภายในโปรแกรม คือการดูว่าทุกอินพุตที่เข้าโปรแกรมจะต้องถูกนำไปใช้และได้ผลลัพธ์ออกมา (หรือถูกลบไป) ส่วนการกำหนดส่วนควบคุมระหว่างโปรแกรมจะดูว่าทุก ๆ เอาต์พุตของโปรแกรมหนึ่งจะต้องถูกนำไปใช้งาน

2.2 ออกแบบฐานข้อมูล และสร้างกระบวนการอัตโนมัติ

วัตถุประสงค์ของการออกแบบฐานข้อมูล ก็เพื่อเปลี่ยนแบบจำลองข้อมูลที่ได้มาจากการ "วิเคราะห์" ให้ไปเป็นโครงสร้างข้อมูล ที่สามารถใช้ระบบจัดการฐานข้อมูล (Database Management System) หรือระบบจัดการไฟล์อื่น ๆ ได้ เพื่อให้ผู้พัฒนาแอพพลิเคชันจะได้เห็นและให้สามารถกำหนดเครื่องดิสก์ และโครงสร้างการใช้งานข้อมูลได้ นอกจากนั้นยังเพื่อรองรับประสิทธิภาพและความน่าเชื่อถือด้วย

วิธีการออกแบบฐานข้อมูลก็เริ่มจากทบทวนสถาปัตยกรรมแอพพลิเคชัน การกระจายข้อมูลและมาตรฐานต่าง ๆ เลือกซอฟต์แวร์จัดการฐานข้อมูล (หรือไฟล์ธรรมดา) สำหรับข้อมูลแต่ละชุด ออกแบบโครงสร้างของ DBMS แบบเชิงสัมพันธ์ (relational) โดยการกำหนดตาราง (table) คอลัมน์ (column) และคีย์หลัก (primary key) ของตาราง ออกแบบคีย์อื่น ๆ ถ้ามี เช่น foreign key, ออกแบบ view และทำ denormalize ตามสมควร จากนั้นจึงทำการออกแบบหรือเลือกระบบจัดการฐานข้อมูลเชิงกายภาพ

แล้วจึงทดสอบฐานข้อมูลที่มีกับมาตรฐานที่ใช้ (benchmark) และปรับปรุงแก้ไขจนได้ตามเกณฑ์ จากนั้นจึงไปทบทวนงานออกแบบฐานข้อมูลนั้นกับผู้ดูแลระบบฐานข้อมูลในองค์กรนั้น

การเลือกซอฟต์แวร์จัดการฐานข้อมูลบ่อยครั้งก็มักจะเลือกตัวที่ใช้อยู่ เพื่อความสะดวกในการเชื่อมต่อ แต่ก็ไม่ได้จำเป็นอะไร

วัตถุประสงค์ของการสร้างกระบวนการอัตโนมัติ ก็เพื่อออกแบบทุกโปรแกรมให้ถึงระดับที่นำมาโค้ดได้ โดยสามารถประมาณการใช้ทรัพยากรของเครื่องและเวลาในการตอบสนองได้พอสมควร เพื่อออกแบบส่วนเชื่อมต่อระหว่างแอพพลิเคชันให้อัตโนมัติและสอดคล้องกัน และเพื่อออกแบบการนำข้อมูลเก่าที่มีอยู่มาใส่ในระบบใหม่ซึ่งจะเกิดขึ้นในส่วน "การนำไปใช้"

วิธีการเริ่มจากการแบ่งแต่ละโปรแกรมให้เป็นงานย่อย ๆ (module) กำหนดวัตถุประสงค์และวิธีการทำงานของแต่ละงานย่อย (module) ออกแบบส่วนเชื่อมต่อระหว่าง module ออกแบบรูปแบบการใช้งานฐานข้อมูล (database access) สำหรับส่วนเชื่อมต่อแต่ละอัน กำหนดชนิดการเชื่อมต่อ (ไฟล์ เมสเสจ เป็นต้น) รูปแบบและการควบคุม

ระดับของการแบ่งย่อยงานคงต้องอาศัยประสบการณ์อยู่บ้าง แต่ข้อคำนึงก็มีเช่น แต่ละ module ควรจะมีวัตถุประสงค์ที่ชัดเจนอย่างใดอย่างหนึ่ง และกำหนดความสัมพันธ์กับ module อื่นอย่างชัดเจน เช่น อาจจะให้บริการการพิมพ์แบบแนวตั้ง (portrait) สำหรับฐานข้อมูลอันหนึ่ง ให้ module อื่นส่งชื่อฐานข้อมูลมาให้ และอาจตามด้วยข้อกำหนดเพิ่มเติมหรือพารามิเตอร์จำนวนหนึ่ง

ข้อคำนึงในการออกแบบมีอยู่มากมาย สำหรับข้อคำนึงในแง่ของการบำรุงรักษาในอนาคตก็จะต้องให้แต่ละ module ระดับของความเกี่ยวพัน (coupling) ต่ำ คืออิสระจากกัน เพื่อเพิ่มโอกาสที่จะเรียกใช้ได้จาก module ระดับบนกว่าได้หลาย ๆ module และช่วยให้เกิดผลข้างเคียง (side effect) น้อยลงในเวลาที่ต้องปรับแก้ไข module ใด module หนึ่ง วิธีการ

ลดความเกี่ยวพันกันก็เช่น ลดจำนวนการเชื่อมต่อลง ทำให้การเชื่อมต่อเป็นไปอย่างง่ายดายและเป็นธรรมชาติ และลดความสัมพันธ์ระหว่างโมดูลที่ไม่จำเป็นลง

นอกจากความเกี่ยวพัน (coupling) แล้วยังมีเรื่องของ ความยึดเหนี่ยว (cohesion) ซึ่งหมายถึง ความแข็งแกร่งของความสัมพันธ์กันขององค์ประกอบภายในโมดูล cohesion ยิ่งสูงยิ่งดี cohesion จะสูงสุดหากทุกองค์ประกอบในโมดูลร่วมกันเพื่อทำงานหนึ่งงานเดียวเท่านั้น ระดับ cohesion ที่ดีจะหมายถึง โมดูลนั้นมีวัตถุประสงค์ชัดเจนว่าจะทำอะไรให้แก่โมดูลที่อยู่เหนือขึ้นไปและเรียกใช้โมดูลนั้น

ข้อคำนึงสำหรับการออกแบบกระบวนการอัตโนมัติในแง่ของประสิทธิภาพก็เพื่อให้มีเวลาตอบสนอง (response time) และใช้ทรัพยากรในระดับที่ต้องการ การเพิ่มประสิทธิภาพอาจทำได้โดย ลดขนาดและจำนวนครั้งของการส่งผ่านข้อมูล เช่น ระหว่างไคลเอนต์กับเซอร์ฟเวอร์ ลดการเข้าถึงฐานข้อมูล เช่น ทำ denormalization โครงสร้างข้อมูล หรือการใช้ดัชนีรอง (secondary index) ในโปรแกรมพวกแบตช์ ให้ความสำคัญแก่การใช้ทรัพยากรอย่างมีประสิทธิภาพมากกว่าเวลาตอบสนองที่สั้น เช่น พยายามอ่านฐานข้อมูลเท่าที่จำเป็น อ่านแต่ละรายการ (record) ทำการคำนวณทั้งหมดแล้วค่อยอ่านรายการถัดไปแบบต่อเนื่อง (sequential) เป็นต้น นอกจากนั้นอาจใช้เทคนิคการทดเลข คือคำนวณค่าที่ซับซ้อนมาก ๆ และใช้หลายครั้งขึ้นมาครั้งเดียวแล้วเก็บบันทึกไว้เพื่อลดเวลาการใช้งานซีพียู เป็นต้น

การเตรียมการเพื่อพัฒนาระบบในช่วงถัดไป ก็ต้องจัดทำขึ้น เช่น เตรียมมาตรฐานในการเขียนโปรแกรม เช่น template และพวกคอมโพเนนต์ต่าง ๆ การออกแบบต้องกำหนดรูปแบบของการเขียนโปรแกรมหรือสร้าง template ขึ้นมาให้ใช้ คอมโพเนนต์หมายถึงส่วนของโปรแกรมหรืองานออกแบบที่มีอยู่ซึ่งสามารถนำมาใช้ใหม่ได้ พวกนี้จะมีคำอธิบายประโยชน์ วิธีการใช้งาน และการจัดการข้อผิดพลาดอย่างครบถ้วน ท้ายที่สุดก็เพื่อให้การพัฒนาระบบทำได้รวดเร็วขึ้น

2.3 เขียนโปรแกรมส่วนต่าง ๆ และเตรียมกรณีทดสอบ

วัตถุประสงค์ของการเขียนโปรแกรมส่วนต่าง ๆ ก็เพื่อยืนยันว่าได้งานที่ตอบสนองต่อความต้องการทั้งด้านการใช้งานและด้านคุณภาพที่กำหนดไว้ หลักการก็ได้แก่ ตรวจไล่ดูโปรแกรมและข้อมูลต่าง ๆ กับความต้องการที่กำหนดไว้ก่อนแล้วว่าถูกต้องสมบูรณ์แค่ไหน ได้ตามเกณฑ์มาตรฐานที่ตั้งไว้หรือยัง อาจต้องมีการปรับสเปกที่มีอยู่ได้หากจำเป็น และก็ต้องบันทึกการตรวจสอบหรือแก้ไขทั้งหมดไว้ด้วย

วัตถุประสงค์ของการเขียนโปรแกรมส่วนต่าง ๆ ก็เพื่อให้ได้มาซึ่งโค้ดสำหรับแต่ละส่วนของโปรแกรมนั่นเอง วิธีการก็เริ่มจากทบทวนดูสเปกที่มี เขียนโปรแกรมส่วนต่าง ๆ ให้เสร็จ และบันทึกกรณีที่จะไว้ทดสอบด้วย ทดสอบและแก้ไขโค้ดโปรแกรมแต่ละส่วนให้ถูกต้อง แล้วให้เพื่อนทดสอบดู เทคนิคในการเขียนโปรแกรมมีอยู่มากมาย เช่น การใช้ส่วนโปรแกรมที่มีอยู่คล้าย ๆ กันมาแก้ไข การใช้ CASE ทูลต่าง ๆ การใช้ Object Orientation หรือใช้ภาษาชั้นสูง (พวก 4GL เป็นต้น) ก็เพื่อให้ได้โค้ดที่ถูกต้องอย่างรวดเร็ว

วัตถุประสงค์ของการเตรียมกรณีทดสอบ ก็เพื่อเตรียมเงื่อนไขในการทดสอบ (test condition) ข้อมูล จำนวนรอบการทดสอบ (cycle) และผลลัพธ์ที่คาดหวัง ทั้งหมดนี้ก็เพื่อทดสอบส่วนต่าง ๆ ของระบบ ขั้นตอนก็เริ่มจากทบทวนรูปแบบการทดสอบที่จะใช้ แจกแจงกรณีต่างๆ ที่จะทดสอบโดยการเตรียมเงื่อนไขทดสอบ รอบการทดสอบ กระดาษบันทึก สร้างข้อมูลที่จะใช้ทดสอบและหาผลลัพธ์ที่คาดหวัง แล้วจึงทบทวนกรณีต่าง ๆ เหล่านั้น

ผู้ใช้จะมีส่วนสำคัญด้วยในการช่วยสร้างเงื่อนไขทดสอบและผลลัพธ์ที่คาดหวัง รอบการทดสอบสามารถกำหนดได้โดยรวบรวมเงื่อนไขทดสอบต่าง ๆ เข้าด้วยกัน ในช่วงแรก ๆ จะทดสอบหลักการใช้งานและคุณภาพทั่ว ๆ ไป ในรอบหลัง ๆ ควรจะทดสอบกรณีผิดพลาดต่าง ๆ ให้ครบถ้วนด้วย การทดสอบโปรแกรมพวกไคลเอนต์สามารถทำได้จากการทดสอบการทำงานหรือปฏิกิริยาของชิ้นส่วนต่าง ๆ ของส่วนติดต่อผู้ใช้ เช่น ปุ่ม หน้าต่าง ข้อความต่าง ๆ เป็นต้น ส่วนการทดสอบโปรแกรมเซอร์ฟเวอร์ก็ทำโดยการป้อนอินพุตเข้าไปแล้วตรวจดูการทำงานหรือผลลัพธ์ที่ได้จากเซอร์ฟเวอร์

การเตรียมข้อมูลทดสอบ จะต้องเตรียมข้อมูลมากเพียงพอที่จะครอบคลุมกรณีที่เป็นไปได้ รวมทั้งกรณีผิดพลาด โดยต้องทดสอบทั้งที่ขอบเขต (boundary) และในช่วง (range) อย่างเช่น หากต้องการทดสอบ (x-2)/(x+3) ก็ต้องทดสอบค่า x ที่ 2, -3 และเลือกตัวแทนของทุก ๆ ช่วง เช่น 5, 0, -4 เป็นต้น ทั้งหมด 5 กรณี หลังจากเตรียมข้อมูลทดสอบแล้วก็ต้องเตรียมผลลัพธ์ที่คาดหวังก่อนที่จะทำการทดสอบ

บ่อยครั้งที่เราอาจต้องสร้างโมดูลแทนชั่วคราวเพื่อใช้ทดสอบโมดูลที่ต้องการ โมดูลแทนนี้ทางเทคนิคมี 2 ประเภทคือ Driver Module คือ โมดูลที่เราสร้างขึ้นเพื่อใช้เรียก เพื่อทดสอบโมดูลที่เรากำลังทดสอบ เช่น หากเราอยากทดสอบโมดูลสำหรับการพิมพ์ เราอาจต้องสร้าง Driver สักตัวหนึ่งมาเรียกใช้โมดูลการพิมพ์ อีกประเภทคือ Stub Module คือ โมดูลที่สร้างเพื่อให้ถูกเรียก เพื่อทดสอบโมดูลที่เรากำลังทดสอบ เช่น หากเรากำลังจะทดสอบโมดูลคำนวณเลขชุดหนึ่ง เราอาจต้องสร้าง Stub ขึ้นมา เพื่อให้โมดูลคำนวณของเราเรียกใช้เพื่อแสดงผลลัพธ์จากการคำนวณ

เทคนิคในการทดสอบโปรแกรมอย่างมีแบบแผนยังมีอีกมาก ซึ่งไม่อาจที่จะกล่าวไว้ ณ ที่นี้ ภายใต้เวลาและพื้นที่จำกัด ตัวอย่างเทคนิคก็เช่น การตรวจดู cross-reference ที่คอมไพเลอร์สร้างให้ การตรวจดูคำเตือนจากคอมไพเลอร์ การทดสอบลูป (loop) การเปรียบเทียบ, I/O และอื่นๆ

2.4 ทดสอบโปรแกรมส่วนต่าง ๆ และส่วนเชื่อมต่อ

วัตถุประสงค์ของการทดสอบนี้ก็เพื่อยืนยันความถูกต้องของส่วนโปรแกรมต่าง ๆ ว่าตรงตามความต้องการทั้งหมดที่กำหนดไว้ก่อนหน้านี้หรือไม่

วิธีการคือ เริ่มจากทบทวนวิธีการและเป้าหมายของการทดสอบที่วางไว้ ทำการทดสอบรอบการทดสอบ (cycle) ตรวจสอบผลที่ได้จริงจากการทดสอบกับผลลัพธ์ที่คาดหวัง ปรับแก้ไขและทดสอบใหม่ ทำการทดสอบจนครบโดยไม่ลืมที่จะคอยตรวจดูประสิทธิภาพการทำงาน (performance) ของระบบ

การทดสอบเริ่มจากการทดสอบส่วนย่อยต่าง ๆ ซึ่งเรียกว่า Unit Test เพื่อพิสูจน์ว่าแต่ละส่วนย่อยทำงานได้ถูกต้อง เป้าหมายครบถ้วนหรือไม่ โดยต้องทำการทดสอบทุกเงื่อนไขที่มี หลังจาก Unit Test ก็เป็น String Test ซึ่งเป็นการทดสอบหลาย ๆ โมดูลที่มาเชื่อมต่อกัน เป็นโมดูลที่ใหญ่ขึ้น เราจะทดสอบว่ามันติดต่อกับกลุ่มโมดูลอื่นได้ถูกต้องครบถ้วนแค่ไหน โมดูลต่าง ๆ เหล่านี้จะเชื่อมต่อกันเข้าและใหญ่ขึ้นเรื่อย ๆ จนเป็นโปรแกรม แล้วโปรแกรมต่าง ๆ จะต้องถูกทดสอบการติดต่อถึงกันด้วยเช่นกัน การทำ String Test นี้ บางคนก็เรียกว่า Integration Test นั่นเอง

บางครั้งระหว่างการทำ String Test เราอาจพบข้อผิดพลาดภายในโมดูลที่อาจเกิดจากสเปกที่คลาดเคลื่อนก็เป็นได้ จึงต้องมีการทบทวนสเปกของโมดูลนั้น แก้ไขข้อบกพร่องทำการ Unit Test โมดูลนั้นแล้วจึงต่อ String Test และในบางครั้งการแก้ไขสเปกของโมดูลอาจจะไปกระทบโมดูลอื่น จึงเป็นการยืนยันอันหนึ่งของหลักการออกแบบที่ดีว่าควรมี Coupling ต่ำ จะทำให้โอกาสเกิดผลกระทบเช่นนี้ลดลง

ความจริงบ่อยครั้งจะพบว่า การเขียนโปรแกรมและการทดสอบโปรแกรม จะดำเนินไปพร้อม ๆ กัน เนื่องจากสมัยนี้เครื่องไม้เครื่องมือในการทดสอบมักจะเป็นส่วนหนึ่งของเครื่องมือการพัฒนาหรือเขียนโปรแกรมนั่นเอง

2.5 ทดสอบกับผู้ใช้และความต้องการเชิงคุณภาพ

วัตถุประสงค์ในการทดสอบกับผู้ใช้ก็เพื่อตรวจดูว่า ความต้องการที่กำหนดไว้ในตอนแรกได้รับการตอบสนองครบถ้วนหรือไม่ วิธีการเริ่มต้นจากการทบทวนแผนงานและวัตถุประสงค์และตรวจสอบสิ่งที่จำเป็นต่าง ๆ ทำการทดสอบโดยการรัน ตรวจสอบผลลัพธ์ ประสิทธิภาพและการใช้ทรัพยากร ทำการแก้ไขและทดสอบใหม่ถ้าจำเป็น

การทดสอบนี้จะทำโดยผู้ที่จะใช้ระบบจริง ๆ ภายหลัง โดยทีมที่พัฒนาจะเพียงแค่ให้คำแนะนำและชี้ทางเท่านั้น ผู้ทดสอบควรต้องทดสอบทุกการทำงานเหมือนกับการใช้จริง ๆ โดยต้องแบ่งเวลามาสร้างความคุ้นเคยกับระบบใหม่นี้ด้วย การทดสอบนี้ก็ควรจะใช้ข้อมูลจริงและทดสอบที่เรียกว่า Stress Test เพื่อพิสูจน์ประสิทธิภาพและความถูกต้องเมื่อนำไปใช้จริงว่าจะสามารถรับปริมาณงานเพียงพอหรือไม่ เช่น ทดสอบโปรแกรมที่เป็นเซอร์ฟเวอร์ว่าสามารถรับงานจากไคลเอนต์จำนวนมาก ๆ ที่เหมาะสมค่าหนึ่งได้หรือไม่ การทำ Stress Test ไม่ควรทิ้งไปไว้หลัง ๆ เนื่องจากเป็นการเสี่ยงเกินไป อาจต้องทำการแก้ไขอย่างมากมายหากผลการทำ Stress Test มีปัญหา

การทดสอบเชิงคุณภาพด้านประสิทธิภาพ อาจทำได้ 2 ลักษณะคือ การวิเคราะห์ และการจำลองสถานการณ์ (Simulation) การวิเคราะห์ทำได้โดยการคำนวณว่าทรัพยากรแต่ละอย่างของระบบจะถูกใช้ไปแค่ไหน ทำการคำนวณว่ารับงานได้แค่ไหน ความเร็วของการประมวลผลและของการสื่อสารแค่ไหน เป็นต้น ส่วนการจำลองสถานการณ์ จะวัดแอพพลิเคชันตามเกณฑ์ที่มีโดยการทดลองใช้งานตามตัวอย่างของธุรกรรมและฐานข้อมูล ซึ่งมักจะใช้บ่อยหากสิ่งที่สร้างขึ้นมานั้นเป็นความคิดใหม่ ที่ยังไม่มีข้อมูลให้วิเคราะห์เปรียบเทียบเพียงพอ

3. การนำไปใช้

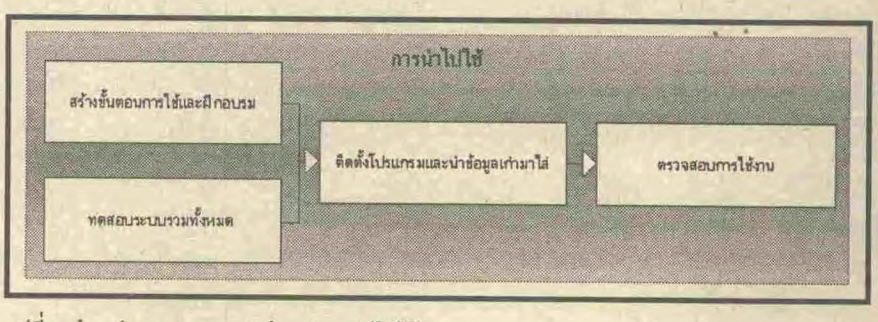
วัตถุประสงค์เพื่อสร้างขั้นตอนการใช้งานสำหรับผู้ใช้และอุปกรณ์ฝึกอบรม เพื่อเตรียมพร้อมสถานที่และผู้ใช้สำหรับการนำระบบใหม่ไปใช้ เพื่อนำระบบทั้งหมดที่จำเป็นเข้าสู่การใช้งานจริง (production) และช่วยเหลือทีมสนับสนุนสำหรับระบบใหม่

ในส่วนนี้จะเตรียมคู่มือการใช้งานและเปลี่ยนข้อมูลเก่าเข้าสู่ระบบใหม่ทั้งหมด ฝึกอบรมการใช้งานและทีมสนับสนุน โดยทั่วไปหากต้องลงแอพพลิเคชันที่หลายแห่ง ส่วนที่ 3 นี้มักจะครอบคลุมเพียงแค่สถานที่ (site) แรกเท่านั้น ส่วนการลงที่อื่น ๆ จะเป็นโครงการต่อเนื่องต่างหากออกไป

ส่วน "การนำไปใช้" อาจจะแบ่งออกไปเป็น 4 งานดังในรูปที่ 5 คือ

1. สร้างขั้นตอนการใช้และฝึกอบรมผู้ใช้
2. การทดสอบระบบรวมทั้งหมด ซึ่งมักจะทำไปได้พร้อม ๆ กันกับข้อ 1.
3. ลงโปรแกรมและนำข้อมูลเก่ามาใส่
4. ตรวจสอบการใช้งานจริง

ก่อนที่จะลงในรายละเอียดจะขอสรุปสิ่งที่ต้องใช้และผลลัพธ์ที่ได้จากส่วน "การนำไปใช้" ก่อน สิ่งที่ต้องใช้แน่นอนย่อมเป็นแอพพลิเคชันที่ผ่าน "การสร้าง" มาแล้ว รวมทั้งผังอธิบายการเชื่อมต่อของหน้าต่างต่าง ๆ และ



รูปที่ 5 ส่วนประกอบของงานส่วน "การนำไปใช้"

ผังอธิบายงานหรือกระบวนการทางธุรกรรมนั้น ๆ ผลลัพธ์ที่เราจะได้ได้แก่ คู่มือการใช้งานและอุปกรณ์การฝึกอบรม (เช่น หนังสือ วิดีโอเทป หรือโปรแกรมสอนทางคอมพิวเตอร์)

3.1 สร้างขั้นตอนการใช้และฝึกอบรมผู้ใช้

เพื่อทำคู่มือและเอกสารประกอบการใช้งานและการสนับสนุนแอพพลิเคชัน และเพื่อฝึกผู้ฝึกสอนที่จะสอนผู้ใช้ต่อไปในอนาคต การทำคู่มือจะเริ่มจากรวบรวมเอกสารต่าง ๆ กำหนดและสร้างขั้นตอนต่าง ๆ เพิ่มเติม ทำเป็นรูปเล่ม (กรณีเป็นหนังสือ) และทบทวนดู คู่มือจะมีสองประเภท คือ สำหรับผู้ใช้งาน และสำหรับผู้สนับสนุน (Support personnel) คู่มือสำหรับผู้สนับสนุนจะมีรายละเอียดทางเทคนิคเพียงพอถึงการทำให้ระบบทำงานได้ และการแก้ไขปัญหาเบื้องต้นที่ผู้ใช้ทั่วไปต้องการความช่วยเหลือ คู่มือในการบำรุงรักษาจะประกอบด้วย บทนำ รายการเอกสารคู่มือเกี่ยวข้องกับระบบ สถานที่ของคู่มือผู้ใช้ ไฟล์สำหรับทดสอบและผลลัพธ์ที่คาดหวัง เครื่องลูกข่ายและเซอร์ฟเวอร์ทั้งหมด รวมทั้งขั้นตอนการบำรุงรักษาอย่างครบถ้วน และไม่ควรลืมรายชื่อและวิธีติดต่อ ผู้ติดต่อสำคัญในกรณีเกิดข้อสงสัยหรือมีคำชี้แนะอะไรด้วย

3.2 ทดสอบระบบรวมทั้งหมด

เพื่อตรวจสอบว่าทุกโปรแกรมสัมพันธ์กันได้ถูกต้อง วิธีการก็เริ่มจากทบทวนแผนงานและวัตถุประสงค์ ตรวจดูทรัพยากรที่ต้องใช้ สร้างรอบการทดสอบ (Test cycle) แล้วทำการทดสอบ ตรวจสอบผลลัพธ์ แก้ไขและทำการทดสอบใหม่ตามจำเป็น อย่าลืมตรวจดูประสิทธิภาพด้วย

การทดสอบนี้เปรียบเสมือนปลายทางของการทดสอบในช่วง "การสร้าง" นั่นเอง หากต้องทดสอบแบบหลายสถานที่ (multiple-site) ก็ต้องทำการทดสอบเพิ่มเติมอีก การทดสอบระบบรวมทั้งหมดนี้ ควรทดสอบส่วนเชื่อมต่อ (interface) ต่าง ๆ ทั้งหมดที่มีในระบบ เช่น การเชื่อมต่อกันของแต่ละแอพพลิเคชัน

3.3 ติดตั้งโปรแกรมและนำข้อมูลเก่ามาใส่

วัตถุประสงค์เพื่อติดตั้งโปรแกรมและลงข้อมูลจริงสู่เครื่องที่จะใช้งานจริง ขั้นตอนเริ่มจากการเตรียมการโดยการทบทวนแผนงาน วัตถุประสงค์ วิธีการนำข้อมูลเข้าระบบ และตรวจสอบสิ่งจำเป็นทั้งหลาย เช่น เครื่อง สถานที่ และข้อมูล จากนั้นจึงทำการเปลี่ยนหรือนำข้อมูลและตรวจสอบดูความถูกต้องครบถ้วน แล้วจึงถอดระบบเก่าที่ถูกแทนที่ด้วยระบบใหม่ และถ่ายทอดหน้าที่รับผิดชอบในการดูแลระบบ

ก่อนจะถอดระบบเก่าออก ควรจะต้องใช้งานระบบใหม่ไปสักระยะหนึ่งเพื่อความมั่นใจ ทั่วไปจะมีการรันสองระบบขนานกันสักระยะหนึ่ง ทั้งนี้ต้องพิจารณาผลกระทบจากการรันขนานนี้ให้รอบคอบก่อนลงระบบใหม่ด้วย การรันระบบเก่าขนานไว้ชั่วคราวก็เพื่อป้องกันความเสียหายที่อาจเกิดขึ้นจากระบบใหม่ ถ้ามีจะได้หันกลับมาใช้ระบบเก่าได้ต่อเนื่องระหว่างปรับแก้ระบบใหม่

ในช่วงติดตั้งนี้ การสื่อสารระหว่างทีมสนับสนุนและผู้ใช้เป็นสิ่งจำเป็น เพื่อความต่อเนื่องของระบบ ได้แก่ การแนะนำอบรมผู้ใช้ถึงระบบใหม่และวิธีการใช้งานและติดต่อประสานงาน การแต่งตั้งผู้ติดต่อถาวรสำหรับระบบ (contact point) และการจัดทำขั้นตอนในการตรวจทาน การบริการผู้ใช้เป็นระยะๆ

3.4 ตรวจสอบการใช้งานจริง

วัตถุประสงค์เพื่อความมั่นใจว่าระบบใหม่ทำงานได้ถูกต้องตามความต้องการ และอย่างมีประสิทธิภาพ และเพื่อบันทึกสิ่งที่ควรปรับปรุงในระบบใหม่นี้ ขั้นตอนก็ต้องสนับสนุนการใช้งานไปในช่วงแรก ตรวจดูความก้าวหน้าของการทำงานที่เปลี่ยนไป บันทึกสิ่งที่ถูกขอให้ปรับเปลี่ยนไว้ให้หมด และปรับแต่งรายละเอียดตามจำเป็น

การตรวจดูการใช้งานระบบใหม่ ควรต้องตรวจว่า ผลลัพธ์ทั้งหมดของระบบครบถ้วนถูกต้อง ตรวจพบข้อผิดพลาดของโปรแกรมทั้งหมดและแก้ไขให้ถูกต้อง ตรวจดูประสิทธิภาพและปรับแต่งให้ดีขึ้น และผู้ใช้งานและทีมดูแลระบบให้สามารถใช้ระบบใหม่ได้อย่างเหมาะสม

อย่าลืมเข้มงวดกับวันเวลาที่กำหนดไว้ ความล่าช้าสามารถเกิดได้จาก ความไม่มีประสิทธิภาพของระบบหรือขั้นตอนการใช้งาน หรือการดูแลระบบ จากการกำหนดระยะเวลาที่ไม่เหมาะสม จากความบกพร่องของเครื่องไม้เครื่องมือหรือการส่งมอบงาน

4. การจัดการ

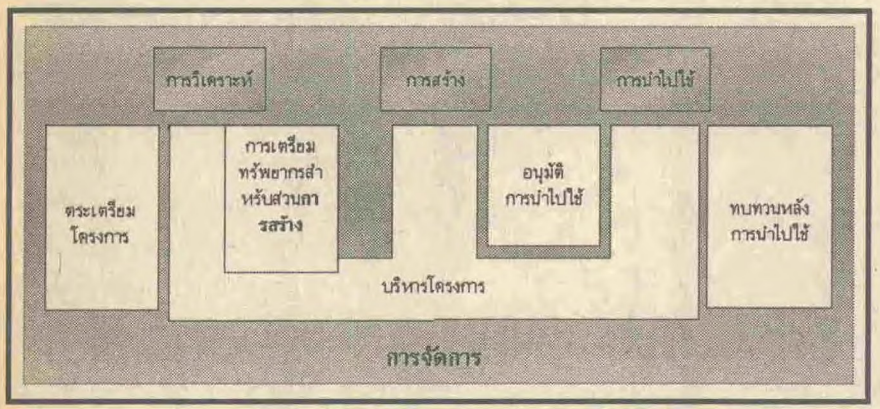
วัตถุประสงค์ของ "การจัดการ" โครงการ RAD ก็เพื่อ

1. สนับสนุนและช่วยเหลือให้โครงการสำเร็จลุล่วงตามเป้าหมาย
2. เพื่อประสานความร่วมมือและการอนุมัติจากผู้ใช้
3. เพื่อวัดมาตรฐานและเก็บบทเรียนที่ได้จากการทำโครงการเพื่อประโยชน์ในอนาคต

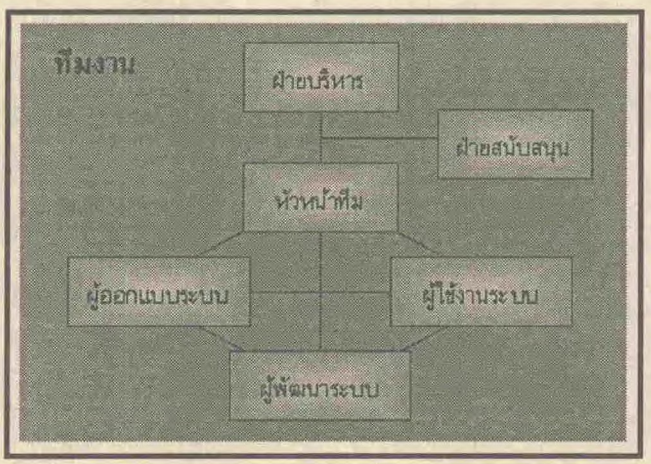
การจัดการและบริหารโครงการแบบ RAD นี้สำคัญมาก เนื่องด้วยระยะเวลาที่จำกัด ผู้จัดการโครงการมีหน้าที่รับผิดชอบสนันสนุนให้โครงการก้าวหน้า ตรวจสอบการใช้ทรัพยากรให้มีประสิทธิภาพสูงสุด ประสานบุคคลต่าง ๆ ทั้งภายในทีมและระหว่างทีมกับผู้ใช้ และต้องแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นได้ทันเวลา เพื่อให้บรรลุวัตถุประสงค์และภายในระยะเวลาของโครงการนั้น ๆ

"การจัดการ" นี้ อาจแบ่งออกได้เป็น 5 ส่วนย่อย ดังในรูปที่ 6 คือ

1. การตระเตรียมโครงการ ซึ่งได้แก่ การวางแผนงานและจัดเตรียมบุคลากรต่าง ๆ ในช่วงแรกสุดของโครงการและกำหนดขอบเขตของงานและระยะเวลา



รูปที่ 6 ส่วนประกอบของงานส่วน "การจัดการ"



รูปที่ 7 โครงสร้างของทีมงานในโครงการ

2. บริหารโครงการ ได้แก่ การดูแลบริหารทีมงานในระหว่างการ "วิเคราะห์" "การสร้าง" และ "การนำไปใช้" รวมทั้งตรวจสอบการใช้ทรัพยากรและแก้ไขปัญหาต่าง ๆ

3. กำหนดเวลาและงบประมาณของรอบการวนซ้ำในส่วน "การสร้าง" เพื่อควบคุมทรัพยากร (เวลาและงบประมาณ) ที่จำกัด

4. อนุมัติการนำไปใช้ เป็นการอนุมัติการจบส่วน "การสร้าง" เพื่อเตรียม "นำไปใช้"

5. ทบทวนหลังการนำไปใช้ เพื่อประเมินประสิทธิภาพของระบบ ทั้งที่วัดเป็นตัวเลขได้และความรู้สึกพึงพอใจของผู้ใช้ นั่นคือระบบนั้นสนองตอบความต้องการของธุรกิจและด้านคุณภาพแค่ไหน

ก่อนที่กล่าวถึงรายละเอียดของ 5 ส่วนย่อยนี้ ลองไปดูสิ่งที่ต้องใช้ในการทำส่วน "การจัดการ" และผลลัพธ์ที่จะได้จาก "การจัดการ" นี้ สิ่งที่ต้องใช้ ได้แก่ ข้อมูลเบื้องต้นเกี่ยวกับองค์กร และเหตุผลด้านต่าง ๆ รวมทั้งด้านการเงินที่จะทำโครงการนี้ และข้อมูลเกี่ยวกับขอบเขต และแนวนโยบายที่เกี่ยวกับโครงการนี้ ผลที่ได้จาก "การจัดการ" นี้ ก็คือข้อสรุปการเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นระหว่างโครงการ เช่น ความต้องการของผู้ใช้ คุณสมบัติและหน้าที่ของระบบ และอื่น ๆ

ต่อไปนี้จะกล่าวถึงรายละเอียดระดับหนึ่งของส่วนย่อยทั้ง 5 ส่วนของ "การจัดการ"

4.1 การตระเตรียมโครงการ

เพื่อเตรียมกลไกต่าง ๆ ที่จะใช้บริหารและควบคุมความก้าวหน้าของโครงการ เพื่อรวบรวมทีมงานด้านต่าง ๆ และเพื่อรับการอนุมัติยินยอมให้เริ่มทำโครงการนี้ ขั้นตอนได้แก่ กำหนดและยืนยันขอบเขตและวัตถุประสงค์ของโครงการ จัดหาผู้สนับสนุนโครงการ (โครงการใดขาดการสนับสนุนที่พอเพียงก็คงจะดำเนินไปไม่ได้) กำหนดตัวบุคคลและบทบาทของทั้งผู้ใช้งานระบบและทีมงานวิเคราะห์และพัฒนาระบบ กำหนดแผนงาน (workplan) และสร้างความเข้าใจเกี่ยวกับโครงการนั้น ๆ แก่ทีมงานและผู้เกี่ยวข้องทั้งหลายให้ตรงกันก่อนเริ่มงาน (คล้ายกับการปฐมนิเทศทีมงานนั่นเอง)

โดยทั่วไปโครงสร้างของทีมงานในโครงการมักจะคล้าย ๆ กับในรูปที่ 7 บุคคลต่าง ๆ จะต้องให้เวลาเต็มที่กับโครงการนี้ งานอื่นควรจะเลื่อนออกไปหรือให้ผู้อื่นรับผิดชอบ หัวหน้าทีมมีหน้าที่ประสานการใช้ทรัพยากรต่าง ๆ ทั้งทีมพัฒนา ผู้ใช้ เครื่องไม้เครื่องมือต่าง ๆ ตามความจำเป็น การอบรมทีมงานถึงนโยบายและการทำงานในลักษณะ RAD ก็เป็นสิ่งจำเป็น อันเนื่องมาจากการเปลี่ยนแปลงที่รวดเร็วของโครงการ ทุกคนควรจะตระหนักถึงเวลาของโครงการที่มีอยู่จำกัด

สิ่งอำนวยความสะดวกในการพัฒนาระบบและบริหารจัดการโครงการ การจัดการนี้ก็ได้แก่ การวางแผนงาน การติดตามและรายงานมาตรฐานและขั้นตอนของการทำงาน สิ่งอำนวยความสะดวกในการทำงานทั่วไป ได้แก่ ห้องหับสถานที่ เฟอร์นิเจอร์ เช่น โต๊ะทำงาน โต๊ะคอมพิวเตอร์ เครื่องคอมพิวเตอร์ และอุปกรณ์ต่อพ่วงต่าง ๆ กระดานเขียน และเครื่องเขียนต่าง ๆ ตามความจำเป็นของการจัดการและการทำงาน (ซึ่งรวมทั้งการประชุมย่อยภายในโครงการด้วย) สิ่งเหล่านี้มีไว้เพื่อส่งเสริมทีมพัฒนาให้ทำงานได้เร็วขึ้น และเพื่อกระบวนการในการตัดสินใจ (เช่น การประชุมการทำรายงาน เป็นต้น)

4.2 บริหารโครงการในระหว่าง "การวิเคราะห์" "การสร้าง" และ "การนำไปใช้"

เพื่อมั่นใจในขอบเขตของโครงการให้สมดุลกับความต้องการของธุรกิจ เพื่อแก้ไขปัญหาหรือข้อขัดข้องต่าง ๆ ภายในโครงการให้ลุล่วงไปตามเป้าหมายของโครงการและขององค์กรอย่างรวดเร็ว เพื่อให้โครงการดำเนินไปอย่างมีประสิทธิภาพและประสิทธิผล เพื่อ

ประสานการติดต่อระหว่างบุคคลต่าง ๆ เช่น ทีมพัฒนากับผู้ใช้ และฝ่ายบริหารขององค์กรกับฝ่ายบริหารของผู้ใช้ และเพื่อจะได้ปิดโครงการได้ทันตามเป้าที่วางไว้

งานที่ต้องทำได้แก่ รวบรวมเกณฑ์มาตรฐานและสถานะต่าง ๆ ของโครงการ มองและแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นหรืออาจจะเกิดขึ้น สื่อสารสถานะของโครงการกับผู้สนับสนุนโครงการ ประสานการใช้ทรัพยากรและบุคคลต่าง ๆ ของโครงการ และประเมินความก้าวหน้าของการทำงานของทีมกับแผนงานที่วางไว้

RAD มักจะประกอบด้วยทีมงานที่มีประสบการณ์ ผู้จัดการโครงการจึงจำเป็นจะต้องประสานฝ่ายต่าง ๆ ที่มีประสบการณ์เหล่านี้ ควบคุมดูแลการใช้ทรัพยากรต่าง ๆ (รวมทั้งเวลา) การจัดการโครงการเป็นส่วนที่จะผลักดันให้โครงการสำเร็จลุล่วงตามวัตถุประสงค์ที่มีอยู่ภายในเวลาที่จำกัด เวลาส่วนใหญ่ (65-70%) จะอยู่ที่ส่วน "การสร้าง" ดังในรูปที่ 8 การปรับเปลี่ยนขอบเขตของงานจะต้องพิจารณาให้รอบคอบ คำนึงถึงลำดับความสำคัญของความต้องการต่าง ๆ รวมทั้งความก้าวหน้าของโครงการและเวลาที่เหลืออยู่ ปัจจัยที่เกี่ยวข้องกับการจัดการโครงการ RAD ได้แก่ เทคนิคจัดการด้านคุณภาพ ได้แก่ การวัดขอบเขตและลำดับความสำคัญของความต้องการ การทุ่มเทกำลังความสามารถและเวลาของผู้จัดการโครงการเพื่อแก้ปัญหาด้านขอบเขตของงาน กระบวนการตัดสินใจที่จะใช้แก้ปัญหาต่าง ๆ ทีมงานขนาดเล็กซึ่งจะลดความคลาดเคลื่อนในการติดต่อสื่อสาร ตัวแทนของผู้ใช้ระบบซึ่งทราบงานเป็นอย่างดี

การบริหารรอบวนซ้ำในส่วน "การสร้าง" จะต้องทำอย่างรอบคอบเป็นพิเศษ แต่ละรอบคล้ายกับโครงการย่อย ๆ อันหนึ่ง ซึ่งมีทรัพยากรและเวลาจำกัด ในบางครั้งอาจจะต้องมีการเปลี่ยนแนวทางการทำงานบ้างหากวิธีทำงานที่ใช้จะทำให้งานไม่สำเร็จตามเป้าหมายที่วางไว้ ความล่าช้าของงานอาจจะเกิดได้จากหลายสาเหตุ เช่น การประเมินโครงการในช่วงแรกคาดการณ์ต่ำเกินไป ขอบเขตโครงการขยายตัวเกินกว่าที่เผื่อไว้ในตอนแรกหรือความไม่พร้อมด้านเวลาของผู้ใช้ระบบ เป็นต้น

การทำรายงานในโครงการแบบ RAD แม้จะไม่สำคัญเท่ากับในโครงการใหญ่ แต่ผู้จัดการโครงการก็ต้องรับผิดชอบในการทำรายงานสถานะของโครงการและรายงานต่อกลุ่มผู้ใช้และผู้สนับสนุนโครงการ รายงานชุดที่ทำให้กลุ่มผู้ใช้ควรจะใช้ศัพท์เทคนิคเท่าที่จำเป็นและอธิบายเพื่อให้เข้าใจง่าย ผู้จัดการโครงการยังต้องมีหน้าที่สนับสนุนทรัพยากรและบุคลากรตามความจำเป็น ทรัพยากรที่ว่านี้ ได้แก่ สถานที่ เครื่องคอมพิวเตอร์ และอุปกรณ์ต่อพ่วงต่าง ๆ

4.3 เตรียมทรัพยากรสำหรับส่วน "การสร้าง"

วัตถุประสงค์เพื่อจัดการขอบเขตและเป้าหมายของการวนซ้ำ (interaction) เพื่อเตรียมวิธีการทดสอบผลที่ได้จากแต่ละรอบ เพื่อปรับขอบเขตให้สอดคล้องกับกำหนดเวลา และเพื่อเตรียมการสำหรับขั้นต่อไป ส่วนนี้เกิดขึ้นในช่วงจบส่วน "การวิเคราะห์" และจะเข้าสู่ส่วน "การสร้าง" งานที่ต้องทำได้แก่ กะประมาณปริมาณงานที่ต้องใช้ในช่วง "การสร้าง" โดยพิจารณาลำดับความสำคัญด้วย พิจารณาขอบเขตโดยรวมทั้งหมดของโครงการ พิจารณาการจัดวางทีมงานสำหรับส่วน "การสร้าง" กำหนดเป้าหมายขอบเขตและวิธีทดสอบของแต่ละรอบการวนซ้ำ และสร้างความเข้าใจและยืนยันแนววิธีและขอบเขตการทำงานต่อทีมงานและผู้สนับสนุนโครงการ

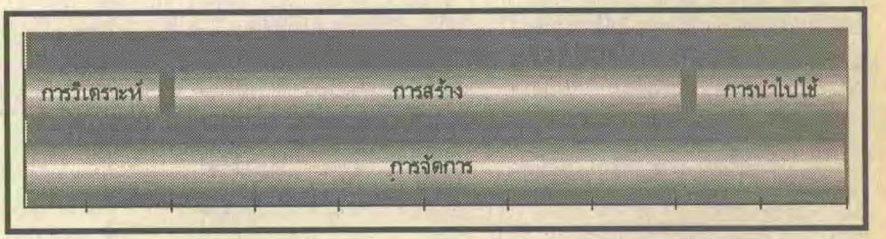
การกำหนดและบริหารรอบการวนซ้ำเป็นสิ่งสำคัญมากในช่วงนี้ การแบ่งรอบวนซ้ำนี้เป็นการแบ่งทอนงานขนาดใหญ่ให้มีขนาดเล็กลงอย่างมีเหตุผล แต่ละรอบจะมีขอบเขตวัตถุประสงค์และวิธีทดสอบผลลัพธ์อย่างชัดเจน

ขั้นตอนในช่วงนี้เริ่มจากทบทวนลำดับความสำคัญและความเสี่ยงของความต้องการต่าง ๆ ของระบบเพื่อพิจารณาขนาดของทีมพัฒนา จากนั้นจึงสร้างกรอบเวลาที่จะใช้พัฒนาระบบตามความต้องการของระบบ โดยให้ได้ผลงานออกมาภายใต้เวลาและทรัพยากรจำกัด ซึ่งจะได้ทราบค่าใช้จ่ายและขอบเขตงานในแต่ละช่วง จากนั้นจึงกำหนดแบ่งย่อยงานออกเป็นรอบ ๆ โดยแต่ละรอบมีจุดหมาย วันเริ่มวันสิ้นสุด และขอบเขตที่แน่ชัด หลังจากนั้นจึงทำการวางแผนเตรียมติดตั้งและแผนการทดสอบ แผนการติดตั้งต้องคำนึงถึงการลงโปรแกรมและข้อมูลต่าง ๆ การอบรมผู้ใช้ และการอบรมการดูแลระบบ แผนการทดสอบจะคำนึงถึงวัตถุประสงค์ในการทดสอบ และขึ้นอยู่กับความสามารถและประสบการณ์ของทีมพัฒนาเป็นอย่างมาก การทดสอบใน RAD ก็เป็นไปในทำนองเดียวกับงานใหญ่ ๆ ซึ่งประกอบด้วยการทดสอบส่วนย่อยต่าง ๆ การเชื่อมต่อกันของส่วนย่อยนั้น ๆ และการใช้งานของผู้ใช้

โดยทั่วไปแล้ว "การสร้าง" มักประกอบด้วยรอบการวนซ้ำประมาณ 3-5 รอบ แล้วแต่ความจำเป็น รอบสุดท้ายมีไว้เก็บกวาดรายละเอียดต่าง ๆ เท่านั้น จะไม่เพิ่มเติมการทำงานใหม่เข้าไป และโครงการส่วนมากต้องมีการปรับขอบเขตของงานโดยเป็นไปได้ทั้งลดและเพิ่มงาน

4.4 อนุมัติการนำไปใช้

วัตถุประสงค์เพื่อกำหนดและยืนยันแผนงาน วิธีการและระยะเวลานำระบบที่สร้างขึ้นไปใช้ เพื่อทบทวนการประเมินองค์ประกอบต่าง ๆ ทางธุรกิจที่เกี่ยวข้องก่อนลงมือนำไปใช้ เพื่อจัดวางทรัพยากรต่าง ๆ เพื่อรองรับการนำ



รูปที่ 8 ช่วงเวลาทั่ว ๆ ไปในโครงการ

ระบบไปใช้จริง และเพื่อกำหนดนโยบายและวิธีการฝึกอบรมที่จะมีขึ้นต่อไป

งานในส่วนนี้ได้แก่ การสรุปการวิเคราะห์ธุรกิจและแผนการนำระบบไปใช้ จัดวางแผนและขั้นตอนที่จะนำระบบไปใช้และสำหรับการฝึกอบรม

โดยทั่ว ๆ ไป นโยบายการนำระบบเข้าใช้งานจริง สำหรับ RAD มักจะมองที่ลำดับของสถานที่จะลงระบบ (site) มากกว่าลำดับของการใช้งาน (function) โดย RAD จะพิจารณาการลงระบบ ณ ที่เดียว ช่วงเวลาที่วางไว้ตั้งแต่ต้นโครงการจะครอบคลุมการลงระบบ ณ แห่งแรกเท่านั้น โดยมักจะไม่รวมการลงระบบ ณ ที่อื่น ๆ เพื่อลดความซับซ้อนของการประเมินโครงการและการกำหนดระยะเวลา ส่วนหากต้องลงมากกว่า 1 แห่ง ก็อาจทำได้โดยวางแผนการติดตั้งเป็น 2 ช่วง คือ ลงที่แรก และลงที่อื่น ๆ ภาระงานการลงทั้ง 2 แบบจะแตกต่างกัน การลงที่แรกต้องการฝึกอบรมและเอกสารมากกว่าการลงที่อื่น ๆ ซึ่งจะเป็นงานที่ซ้ำ ๆ กันไป

4.5 ทบทวนหลังการนำไปใช้

วัตถุประสงค์เพื่อประเมินผลการทำงานและระบบที่ติดตั้งไป หลังจากที่ติดตั้งระบบและใช้งานไปแล้ว ก็ต้องไม่ลืมที่จะรวบรวมข้อมูลและวิเคราะห์ผลของงาน เพื่อจะได้นำไปปรับปรุงพัฒนาขั้นตอนการทำโครงการแบบ RAD ต่อไป โดยทำการประเมินผลการใช้งานจริงที่เกิดขึ้นกับประสิทธิภาพการสร้างระบบที่สนองต่อความต้องการทางธุรกิจ ประสิทธิภาพด้านต้นทุน การเกิดประโยชน์ที่แท้จริง และประสิทธิภาพของขั้นตอนการทำงาน เกณฑ์ที่ใช้วัดจะต้องเที่ยงธรรมและปราศจากความลำเอียงใด ๆ ทั้งสิ้น และจะไม่ใช่เอาไว้เป็นเครื่องมือทำร้ายผู้ใด แต่มีไว้เพื่อพัฒนาการทำงานต่อไปในอนาคตอย่างแท้จริง

ประสิทธิภาพการสนองตอบความต้องการทางธุรกิจอาจจะได้แก่ จำนวนความต้องการที่คาดหวังไว้ในตอนแรกกับจำนวนที่ตอบสนองได้จริง (ไว้ดูการกำหนดขอบเขตของงานและเวลา) และจำนวนความต้องการที่ตอบสนองได้จริงกับจำนวนที่ทำขึ้นและส่งมอบงานจริง ๆ อันนี้เป็นอีกขั้นหนึ่งที่ผู้ใช้ระบบจะพบจริง ๆ ซึ่งสัดส่วนคู่หลังควรจะใกล้เคียงหนึ่งมากที่สุด การสูงกว่าหนึ่งอาจหมายถึงการลดขอบเขตของงาน หรือการประเมินสถานการณ์ต่ำเกินไปในการพิจารณา

การวิเคราะห์การออกแบบระบบและประสิทธิภาพอาจทำได้จากการรวบรวมข้อมูลจากการสัมภาษณ์ผู้ใช้ บันทึกหรือ log ของระบบ เครื่องมือหรือเทคนิควัดประสิทธิภาพเป็นต้น การวิเคราะห์ควรจะพิจารณาว่า วิธีการ (เชิงเทคนิค) ที่ใช้เหมาะสมหรือไม่? ระบบที่ได้สอดคล้องกับความต้องการเชิงคุณภาพแค่ไหน? ขั้นตอน (ทั้งอัตโนมัติและไม่อัตโนมัติ) เหมาะสมและมีประสิทธิภาพหรือไม่? ฮาร์ดแวร์และซอฟต์แวร์ทั้งหลายที่ใช้ทำงานได้ตามที่ผู้ขายกล่าวไว้หรือไม่? ขั้นตอนการทำงานต่าง ๆ และปริมาณงานที่ระบบทำงานเหมาะสมหรือไม่? และผู้ใช้ได้รับบริการความช่วยเหลือต่าง ๆ เหมาะสมหรือไม่?

และควรทำการวิเคราะห์ขั้นตอนที่ใช้ทำ RAD ทั้งหมดนี้ด้วย เพื่อปรับปรุงให้ดีขึ้น ทั้งหมดที่กล่าวมานี้ไม่ใช่คัมภีร์หรือไบเบิลอะไรที่จะถูกต้องร้อยเปอร์เซ็นต์ตลอดไป หากแต่ต้องการการพัฒนาเพิ่มเติม แก้ไข และตัดส่วนที่เกินออกไปอยู่เป็นระยะ ๆ การทบทวนนี้ก็ควรพิจารณาว่า ข้อแนะนำสำหรับการประเมินในขั้นต่าง ๆ เหมาะสมกับงานที่ลงไปแค่ไหน? เอกสารต่าง ๆ ถูกต้องครบถ้วนแค่ไหน? ช่วยลดเวลาการศึกษาเรียนรู้ลงได้แค่ไหน? และขั้นตอนต่าง ๆ สามารถถือเป็นมาตรฐานได้มากน้อยแค่ไหนและประสิทธิภาพเป็นอย่างไร? เป็นต้น

ท้ายสุดก็ควรเขียนรายงานสรุปหลังการนำไปใช้ขึ้นมาให้เรียบร้อยได้มาตรฐาน เนื้อหาควรจะมีกล่าวถึงความเบื้องต้น (overview) ที่จะบรรยายสถานะโดยรวมต่าง ๆ ประสิทธิภาพของระบบที่จะบรรยายว่าระบบทำงานได้ประสิทธิภาพ (ทั้งด้านงานและคุณภาพ) โดยใช้ทรัพยากรต่าง ๆ แค่ไหน ข้อสรุปเชิงธุรกิจที่จะบรรยายผลกระทบด้านต่าง ๆ ต่อธุรกิจ และข้อเสนอแนะเพิ่มเติม ที่จะบรรยายข้อเสนอให้แก้ไขเพิ่มเติมต่าง ๆ ที่ได้รับมา และประเมินทรัพยากร (ทั้งคน เวลา และอื่น ๆ) ที่คาดว่าต้องใช้เพื่อสนองข้อเสนอเหล่านั้น

ทิ้งท้าย

ทั้งหมดนี้เป็นภาพรวมกว้าง ๆ ของการทำโครงการแบบ Rapid Application Development ซึ่งน่าจะพอนำไปใช้ให้เกิดประโยชน์ได้ แต่อย่างไรก็ตาม โครงการจะสำเร็จอย่างดีหรือไม่ก็ขึ้นกับทุกฝ่ายที่เกี่ยวข้องที่เขียนไว้ ณ ที่นี้ก็เป็นเพียงแนวทางปฏิบัติคร่าว ๆ ที่น่าจะลองระลึกถึงเมื่อจะทำโครงการใด ๆ และแนวทางนี้ยังต้องอาศัยการปรับเปลี่ยนพัฒนาให้ดีขึ้นอีกต่อไป

หากมีเวลาและโอกาส ผมจะได้เขียนเพื่อเล่าสู่กันฟังถึงเรื่องราวและรายละเอียดของเทคนิคต่าง ๆ ที่เกี่ยวข้องกับขั้นตอนต่าง ๆ ของโครงการพัฒนาแอพพลิเคชันแบบ RAD ซึ่งยังมีรายละเอียดอีกมากที่ไม่สามารถนำมาลงไว้ในครั้งนี้ได้ครบถ้วน

อ้างอิง

1. Carlo Ghezzi, Meehdi Jazayeri, Dino Mandrioli, Fundamentals of Software Engineering, Prentice-Hall International, 1991.

2. Martin J., Rapid Applications Development, Macmillian Publishing, 1991.

3. Steve McConnell, Rapid Development: Taming Wild Software Schedules, Microsoft Press, 1996.

4. Methodology- Rapid Application Development at http://www.systemhouse.mci.com/shltdemo/tree/d0/14/621.htm.

5. Rapid Application Development (RAD) Methodology at http://sysdev.ucdavis.edu/webadm/document/radmethod.htm.